

PATENT**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

Appellant:	Christopher Peiffer	Confirmation No.	9849
Serial No.:	09/975,286		
Filed:	October 10, 2001	Customer No.:	28863
Examiner:	Hareesh N. Patel		
Group Art Unit:	2154		
Docket No.:	1014-152US01/JNP-0489		
Title:	STRING MATCHING METHOD AND DEVICE		

RESPONSE TO NOTICE OF NON-COMPLIANT APPEAL BRIEF

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450,
Alexandria, VA 22313

Dear Sir:

On December 7, 2007, the Office issued a Notification of Non-compliant Appeal Brief with respect to Applicant's Appeal Brief filed November 20, 2007. Specifically, the Office noted that Applicant failed to summarize independent claim 24. In accordance with MPEP 1205.03 (B), Applicant submits herewith a Summary of the Claimed Subject Matter under separate cover in lieu of an entire new brief.

No fee is due. Please also charge any additional fees that may be required or credit any overpayment to Deposit Account No. 50-1778.

SUMMARY OF THE CLAIMED SUBJECT MATTER

Claim 1

Claim 1 recites a computer-implemented method for comparing an unknown string to a predefined string. Claim 1 requires storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers; the predefined string headers may be stored in memory and/or accessed from a linked database on the network device.

Claim 1 requires receiving, with the network device, a network message. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 of the network device 18 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 1 requires, in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102, e.g., a predefined HTTP header.

Claim 1 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying the unknown string at 102 that is described as typically a header of the received message to be compared with the predefined string selected from the database.

Claim 1 requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. Fig. 6 and pg. 12, ll. 1-4 further describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the unknown header and the predefined

string. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation between segments of the unknown header and segments of the predefined string.

Claim 1 requires performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 1 requires comparing the predefined flag and a result of the bitwise operation to produce an indication for a case-insensitive string match, wherein the indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description states that the comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application on pg. 13 states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 1 requires processing the network message based on the indication of the case-insensitive string match. Claim 1 also requires outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching

module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

Claim 24

Claim 24 recites a method of case-insensitive string matching for use in a computer network. Claim 24 requires storing, on a network device, a plurality of predefined strings, wherein the predefined strings represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database.

Claim 24 requires receiving, with the network device, a network message. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 24 requires selecting one of the plurality of predefined strings stored within the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102, the predefined string being described as an HTTP header.

Claim 24 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying an unknown string at 102 that is described as typically a message header to be compared with the predefined string.

Claim 24 requires performing at least one bitwise exclusive OR operation between characters of the selected predefined string and corresponding characters of the unknown string. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings.

Claim 24 requires performing a bitwise OR operation between a result of the bitwise exclusive OR operation and a predetermined flag. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 24 requires comparing the predetermined flag and a result of the bitwise OR operation to produce a single bit output that indicates whether a case-insensitive match exists between the selected predefined string and the unknown string. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 24 requires processing the network message based on the indication of the case-insensitive match. Claim 24 also requires outputting a response from the network device. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

Claim 25

Claim 25 recites a computer networking device for improving data transfer via a computer network, the device comprising a processor configured to compare a client HTTP header with a known HTTP header. Claim 25 requires storing, on the networking device, a database containing a plurality of known HTTP headers. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 25 requires receiving, with the networking device, a client HTTP header and, in response to receiving the client HTTP header, selecting one of the known HTTP headers stored within the database of the network device. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102 that is typically an HTTP header.

Claim 25 requires performing a bitwise exclusive OR operation on binary representations of the client HTTP header and the known HTTP header selected from the database. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings.

Claim 25 requires performing a bitwise OR operation between a result of the exclusive OR operation and a predetermined flag. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 25 requires comparing the predetermined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the client HTTP header and the selected known HTTP header. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 25 requires processing the network message based on the indication of the case-insensitive match. Claim 25 also requires outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output

Claim 26

Claim 26 recites an article of manufacture comprising a storage medium having a plurality of machine-readable instructions. Claim 26 requires storing, on a network device, a database containing a plurality of predefined strings, wherein the predefined strings stored within the database represent known headers for a network communication protocol. Figs. 1, 2 and 4 show embodiments of a network device 18 and 18' having a

string matching module 24 (FIGS. 2 & 3) that implements string matching methods. See, e.g., pg. 5, ll. 15-16. Pg. 9, ll. 3-11 describes string matching module 24 using a predefined (known) string from a record or a hash table created from previous known headers, and may be stored in memory and/or accessed from a linked database.

Claim 26 requires receiving, with the network device, a network message and, in response to receiving the network message, selecting one of the plurality of predefined strings stored within the database of the network device. Pg. 5, ln. 19 – pg. 6, ln. 4 describe string matching module 24 as transferring Hypertext Transfer Protocol (HTTP) messages between clients 12 and server 14.

Claim 26 requires identifying a portion of the network message as an unknown string for comparison with the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes first identifying a predefined string at 102 that is typically an HTTP header.

Claim 26 requires performing a bitwise exclusive OR operation between an ASCII binary representation of at least a segment of the unknown string and an ASCII binary representation of at least a segment of the selected predefined string. Fig. 6 and pg. 11, ll. 15-20 illustrate and describe a method that includes identifying the unknown string at 102 that is described as typically a message header to be compared with the predefined string.

Claim 26 requires performing a bitwise operation between a predefined flag and a result of the exclusive OR operation. Fig. 6 and pg. 12, ll. 1-4 describe the method as including, at 106, performing at least one bitwise XOR operation on the ASCII binary representations of the strings. As another example, as shown in Fig. 8, at step 218, the method includes performing a bitwise XOR operation on segments of each of the strings. Fig. 8 and pg. 13, ll. 10-12 explain that, if the result of the bitwise exclusive OR operation is not equal to zero, the method continues to step 222 to perform a bitwise OR operation on the result of the XOR operation and a predefined 4-byte flag.

Claim 26 requires comparing the predefined flag and a result of the bitwise OR operation to produce an indication for a case-insensitive string match between the predefined string and the unknown string, wherein the indication for the case-insensitive match indicates whether all characters of the unknown string within the network message match all corresponding characters of the identified predefined string so as to match one

of the known headers of the network communication protocol. Fig. 8 and pg. 13, ll. 10-12 explain that step 224 includes checking if the result of the OR operation is less than or equal to the 4-byte flag (i.e., comparing the predefined flag with the result of the OR operation between the result of the XOR operation and the predefined flag). The description describes that comparison produces an indication for the case-insensitive string match indicates whether all characters of the unknown string within the network message match all corresponding characters of the selected predefined string so as to match one of the known headers of the network communication protocol. Specifically, the application states that if the comparison indicates that the result of the OR operation is less than or equal to the 4-byte flag, step 206 of FIG. 8 finds a segment match and goes to step 208. If not, step 206 does not find a segment match and returns to step 216 with a negative string match.

Claim 26 requires processing the network message based on the indication of the case-insensitive match; and outputting a response from the network device based on the processed network message. Fig. 2 and pg. 6, ll. 10-13 describe the network device as including an ASIC 18o that contains the string matching module 24 configured to implement the methods. According to pg 6, ll. 10-13, ASIC 18o, processor 18m, and memory 18k form a controller 18q configured to process HTTP requests. Pg. 9, ll. 1-3 states that, to process HTTP requests efficiently, string matching module 24 is configured to parse headers by using the described string matching method. According to pg. 7, ll. 3-5, headers (i.e., unknown strings) within the HTTP requests require processing or "parsing" so that an appropriate response to the request can be output.

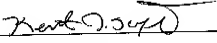
Respectfully submitted,

Date:

By:

January 7, 2008

Shumaker and Sieffert
1625 Radio Drive, Suite 300
Woodbury, Minnesota 55125
Telephone: (651) 286-8343
Facsimile: (651) 735-1102


Name: Kent J. Sieffert
Reg. No.: 41,312